**(A-69989/RMA)**

**HARDWARE ARCHITECTURE NEUTRAL COMPUTER PROGRAM LANGUAGE AND STRUCTURE AND METHOD FOR EXECUTION**

**I Claim:**

1. A hardware architecture neutral executable program structure for execution in a processor, said program structure comprising:

a plurality of instruction threads selected from a library of possible instruction threads;

a plurality of data parameters integrated among at least some of said instruction threads and influencing execution of said instruction threads; and

at least some of said selected instruction threads being adapted for cooperative execution with other of said instruction threads by yielding ownership of said processor upon the occurrence of a predetermined condition.

2. The program structure in claim 1, wherein said instructions comprise operation codes representing commands executable in a processor.

3. The program structure in claim 1, wherein said predetermined condition comprises said yielding instruction yielding after a predetermined time period of ownership.

4. The program structure in claim 1, wherein said predetermined condition comprises said yielding instruction yielding upon determining that a required resource is constrained.

5. The program structure in claims 4, wherein said constrained resource is selected from the group consisting of a memory buffer, an input device, an output device, an input/output device, a digital audio processor, a display device, a communication link, a communication bus, a buffer, a data compression processor, a data decompression processor, a vertical refresh signal (so user does not see display screen refresh), a time limit being exceeded or not yet being exceeded, and combinations thereof.

6. The program structure in claim 5, wherein a characteristic of said constrained resource is the constraining condition associated with the resource.

7. The program structure in claim 6, wherein said characteristics are selected from the group characteristics consisting of: a buffer existing, a buffer not existing, a buffer being initialized, a buffer being uninitialized, a buffer holding a set of data, a buffer not holding a set of data, a buffer holding a subset of a set of data, a buffer not holding a subset of a set of data, and combinations thereof.

8. The program structure in claim 6, wherein said characteristics are selected from the group of an input device, output device, or input/output device signaling that it is available, not available, has text, selection, location, textural or other input data available or not available and combinations thereof.

9. The program structure in claim 6, wherein said characteristics are selected from the group of characteristics consisting of: a digital audio processor, display device, a communication link, a communication bus, a buffer, a data compression processor, a data decompression processor, a vertical refresh signal being in a ready state, a vertical refresh signal not being in a ready state, condition where capacity or features are assured or not assured, and combinations thereof.

10. The program structure in claim 1, wherein said instruction thread is selected from the group of instruction threads that: perform a navigation; make a decision; scale a data item; decompress a data item; set a parameter; use a parameter; circulate a parameter; generate data; generate a parameter or instruction stream; parse a data item; format a data item; select a data item; test a data item; respond to an input; send messages; receive messages; receive responses to messages; request file from a server or other source; store data; perform calculations; perform an animation; perform signal or image processing; respond to a data or command from a user; send a message; request a file; request additional data in a data stream; request data and/or commands in a stream of data and/or commands; navigate; make a decision; scale; decompress; set, use, and calculate parameters; cause audio to be rendered, cause video to be rendered generate other data and/or procedural streams; parse, format, and select text and other media elements such as images, graphics, and audio; respond to item selection by a story player user; request further files during streaming, format XML (or XML extensions); format text; validate user input; perform calculations, simulations, animations, special effects, signal processing, run-time scaling and synchronization tasks; and combinations thereof.

11. The program structure in claim 10, wherein said data items are selected from the set of data items consisting of a digital image media data item, a digital audio media item, transition and special effects control data and combinations thereof.

12. The program structure in claims 10, wherein said response to data or commands, or other input from a user comprises responding by causing a program subroutine or other computer program code to be executed on the thread in which the input, data, or commands are detected.

13. The program structure in claim 10, wherein said requesting additional data and/or commands in a stream of data and/or commands comprises requesting additional ones of said instruction threads integrated with said data parameters.

14. The program structure in claim 1, wherein said cooperative execution is under programmatic control.

15. The program structure in claim 1, wherein:

said predetermined condition is either (i) yielding after a predetermined time period of ownership, or (ii) yielding upon determining that a required resource is constrained, or (iii) a combination of yielding

after a predetermined time period of ownership, and yielding upon determining that a required resource is constrained.

16. The program structure in claim 15, wherein said resource being constrained comprises said resource being unavailable at the time access to said resource is required.

17. The program structure in claim 15, wherein said a predetermined time period of ownership is established programmatically.

18. The program structure in claim 15, wherein said a predetermined time period of ownership is provided as a parameter within said message.

19. The program structure in claim 16, wherein said operation codes comprise integers and an association between said integer and an operation is identified by a table look up procedure, said integers providing a compact representation of said operations.

20. The program structure in claim 1, further including an instruction thread retry attribute associated with at least some of said possible instruction threads, said retry attribute causing said processor to repeatedly retry to execute an instruction thread that has yielded ownership of said processor either (i) after a predetermined time period of ownership, (ii) after running all of the active threads until each has yielded the processor, or (iii) upon determining that a required resource is constrained.

21. The program structure in claim 1, wherein:

said instructions comprise operation codes representing commands executable in a processor;

said predetermined condition comprises said yielding instruction yielding after a predetermined time period of ownership, or said yielding instruction yielding upon determining that a required resource is constrained;

said constrained resource is selected from the group consisting of a memory, an input device, an output device, an input/output device, a digital audio processor, a display device, a communication link, a communication bus, a buffer, a data compression processor, a data decompression processor, a vertical refresh signal (so user does not see display screen refresh), a time limit being exceeded or not yet being exceeded, and combinations thereof; and

said instruction thread is selected from the group of instruction threads that perform a function selected from the set of functions that: perform a navigation; make a decision; scale a data item; decompress a data item; set a parameter; use a parameter; circulate a parameter; cause audio to be rendered; cause video to be rendered; generate data; generate a parameter or instruction stream; parse a data item; format a data item; select a data item; test a data item; respond to an input; send messages; receive messages; receive responses to messages; request file from a server or other source; store data; perform calculations; perform an animation; perform signal or image processing; respond to a data or command from a user; send a message; request a file; request additional data in a data stream;

request data and/or commands in a stream of data and/or commands; navigate; make a decision; scale; decompress; set, use, and calculate parameters; generate other data and/or procedural streams; parse, format, and select text and other media elements such as images, graphics, and audio; respond to item selection by a story player user; request further files during streaming, format XML (or XML extensions); format text; validate user input; perform calculations, simulations, animations, special effects, signal processing, run-time scaling and synchronization tasks; and any combination thereof.

22. A method for cooperatively executing a plurality of code threads in a processor, said method comprising steps of:

(a) communicating a plurality of code threads, including a first code thread and a second code thread, to a processor for execution;

(b) setting a program counter for execution of said first code thread;

(c) allocating ownership of said processor exclusively to execution of said first code thread and executing said first code thread until said first code thread completes execution, except stopping execution of said first code thread and yielding ownership of said processor by said first code thread during said execution to said second code thread upon the occurrence of a predetermined first code thread yield condition;

(d) if execution of said first code thread has been stopped, then storing an indication that execution of said first code thread has been stopped, including a program counter value for said stopped first code thread, in a storage location;

(e) setting said program counter for execution of said second code thread;

(f) allocating ownership of said processor exclusively to execution of said second code thread and executing said second code thread until said second code thread completes execution, except stopping execution of said second code thread and yielding ownership of said processor by said second code thread to any other one of said plurality of code threads upon the occurrence of a predetermined second code thread yield condition;

(g) reallocating ownership of said processor and re-executing said first code thread according to predetermined processor ownership reallocation rules;

(h) retrying execution of said yielded first code thread including setting said program counter with said stored program counter for said stopped first code thread and re-executing said first code thread; and

(i) repeating steps (b) through (g) for each of said plurality of code threads until each of said plurality of code threads has been executed.

23. The method in claim 22, wherein said predetermined first code thread yield condition comprises yielding after a predetermined time period of processor ownership.

24. The method in claim 22, wherein said predetermined first code thread yield condition comprises yielding upon determining that a resource required for execution is constrained.

25. The method in claim 22, wherein said predetermined first code thread yield condition and said second code thread yield conditions are each selected from the group consisting of: (i) yielding after a predetermined time period of ownership, or (ii) yielding upon determining that a required resource is constrained, and a combination thereof.

26. The method in claim 23, wherein said cooperative execution of said plurality of instruction threads is achieved by establishing said predetermined time period of ownership of at least selected ones of said plurality of threads as a instruction thread execution parameter communicated with said instruction thread.

27. A method for cooperatively executing a plurality of code threads in a processor, said method comprising steps of:

sequentially executing a plurality of code threads until a predetermined code thread yield condition is detected for a particular code thread;

stopping execution of said particular code thread for which said thread yield condition was detected;

storing an indication that execution of said particular code thread was stopped before completion in a memory storage location;

resuming sequential execution of said plurality of code threads at the next sequential code thread following said particular code thread;

retrying execution of said particular code thread during said resumed sequential execution according to predetermined rules for preempting a next sequential code thread and retrying execution of said particular code thread in preference to a next sequential code thread.

28. The method in claim 27, wherein said step of retrying includes storing an indicator for said preempted next code thread and retrieving said stored indicator for said particular code thread.

29. The method in claim 28, wherein said stored indicator for said preempted next code thread comprises a program counter value for said preempted next code thread, and said stored indicator for said particular code thread comprises a program counter value for said particular code thread that was yielded.

30. The method in claim 29, further comprising the step of resuming said sequential execution of code threads after said particular code thread has been executed by retrieving said stored program counter value for said preempted next code thread.

31. The method in claim 27, wherein said code thread yield condition comprises yielding after a predetermined time period of processor ownership.

32. The method in claim 27, wherein said code thread yield condition comprises yielding upon determining that a resource required for execution is constrained.

33. The method in claim 27, wherein said predetermined first code thread yield condition and said second code thread yield conditions are each selected from the group consisting of: (i) yielding after a predetermined time period of ownership, or (ii) yielding upon determining that a required resource is constrained, and a combination thereof.

34. The method in claim 27, wherein cooperative execution of said plurality of instruction threads is achieved by establishing said predetermined time period of ownership of at least selected ones of said plurality of threads as a instruction thread execution parameter communicated with said instruction thread.

35. The method in claim 27, wherein cooperative execution of said program instruction threads is achieved by detecting a resource constraint and returning a code to the instruction dispatcher to set the program counter to point back to the same returned instruction before yielding to the next thread.